UNIVERSITY OF EDINBURGH

COLLEGE OF SCIENCE AND ENGINEERING

SCHOOL OF INFORMATICS

AUTOMATED REASONING (LEVEL 10)

AUTOMATED REASONING (LEVEL 11)

Wednesday 1 st May 2013

00:00 to 00:00

Year 4 Courses

Convener: ITO-Will-Determine
External Examiners: ITO-Will-Determine

MSc Courses

Convener: ITO-Will-Determine
External Examiners: ITO-Will-Determine

INSTRUCTIONS TO CANDIDATES

Answer QUESTION 1 and ONE other question.

Question 1 is COMPULSORY.

All questions carry equal weight.

CALCULATORS MAY NOT BE USED IN THIS EXAMINATION

Questions

1. (a) Consider the following NuSMV Finite State Machine model for a simple burglar alarm for a house.

```
MODULE burglar_alarm
  VAR
    state : {off, wait1, live, wait2, triggered};
    set : boolean;
    reset : boolean;
    timeout : boolean;
    door_open : boolean;
  DEFINE
    sound_alarm := (state = triggered);
    run_timer := (state = wait1 | state = wait2);
  ASSIGN
    init(state) := off;
    next(state) :=
      case
        state = off & set       : wait1;
        state != off & reset    : off;
        state = wait1 & timeout : live;
        state = live & door_open : wait2;
        state = wait2 & timeout : triggered;
        TRUE : state;
      esac;
```

Here the `state` variable records the state of the control system, and variables `set`, `reset`, `timeout`, `door_open` model inputs to the system, either from control buttons (the 1st and 2nd), a timer module (the 3rd) or a sensor on the front door of the house (the 4th). The `sound_alarm` signal is an output which causes the alarm to ring, and `run_timer` is an output which when true lets the timer run, and when false resets the timer. The assumption is that the alarm control unit with the set and reset buttons is inside the house, so the timer provides delays allowing the house occupant time to exit after setting the alarm, and time to reset the alarm on entering the house. The timer module is assumed to have similar functionality to that introduced in the second coursework: whenever it is let run, it runs for some number of steps and then asserts a timeout signal.

Draw a finite state automaton model for the system with one state per value of the `state` variable, transitions labelled with appropriate propositional formulas over the input variables, and states labelled with the output signal they make true. Mention any assumptions made about implicit transitions.

[*4 marks*]

(b) Express in Linear Temporal Logic the following properties of the FSM introduced in part (d).

i. It is never the case that the alarm is sounding at the same time as the alarm is switched off.

ii. When the set button is pressed and the reset button is never pressed from that time onwards, the alarm is eventually in the live state.

iii. when the alarm is sounding, the alarm stays sounding up to a time (if any) when the reset button is pressed

[*5 marks*]

(c) The property (e)ii listed above is not satisfied by every run of the model in part (d). Explain why. Give an LTL formula that, when assumed, would make the property (e)ii always true. [*4 marks*]

2. (a) Which of the following pairs of CTL formulas are equivalent? For those which are not, give a model of one of the pair that is not a model of the other – and say which is which. [*7 marks*]

   i. $\mathrm{AG}\,\phi$ and $\mathrm{EG}\,\phi$.

   ii. $\mathrm{AF}\,\mathrm{AX}\,\phi$ and $\mathrm{AX}\,\mathrm{AF}\,\phi$.

   iii. $\mathrm{AF}\,\mathrm{EX}\,\phi$ and $\mathrm{EX}\,\mathrm{AF}\,\phi$.

   (b) i. Briefly explain the concepts of the language $\mathcal{L}(\phi)$ of an LTL formula ϕ and the language $\mathcal{L}(\mathcal{M})$ of a transition system $\mathcal{M}$. [*2 marks*]

   ii. The automata-theoretic approach to LTL model checking phrases the model checking problem $\mathcal{M} \models \phi$ as the language containment problem

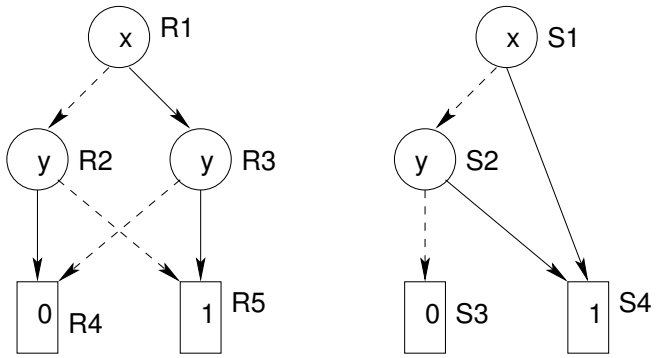
   $$\mathcal{L}(\phi) \subseteq \mathcal{L}(\mathcal{M}) \qquad .$$

   Starting with this observation, give a short high-level description of how automata-theoretic approach to LTL model checking works. There is no need to give formal definitions, instead focus on conveying the key intuitions. [*5 marks*]

   iii. Draw a diagram for an automaton that might be used for model checking the LTL formula F p, assuming the transition system's state is described by boolean-valued variables p and q. State the kind of automaton represented by the diagram. Is the language accepted by the automaton $\mathcal{L}(Fp)$ or the language of some other LTL formula? [*3 marks*]

QUESTION CONTINUES ON NEXT PAGE

(c) i. Given ROBDDs (Reduced Ordered Binary Decision Diagrams) B_f and B_g for boolean formulas f and g, `apply`$(\texttt{op}, B_f, B_g)$ computes an OBDD for f `op` g, where `op` is a symbol for any binary operation on boolean formulas (e.g. $\wedge$, $\vee$, $\oplus$).
Consider ROBDDs B and C



Remember that the dashed edges are for when decision variables are 0 (false) and the the solid edges for when they are 1 (true).
Draw a diagram which illustrates the execution of

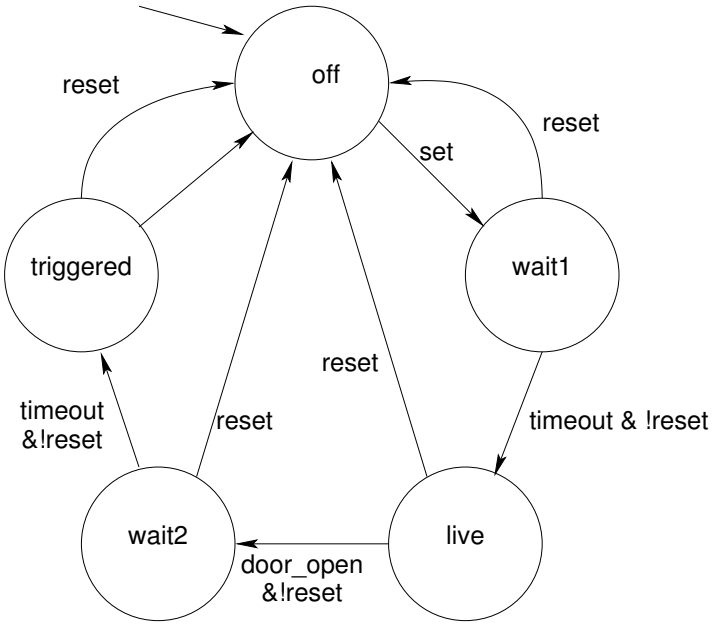
$$\texttt{apply}(\oplus, B, C) \quad ,$$

similar to that shown in the course slides. Remember that $\oplus$ is the exclusive-or operation. The diagram should present the resulting ordered binary decision tree, with each node of the tree annotated with a pair of names of nodes that are the arguments to the `apply` call that created the node. In the course slides, the diagram did not include the terminal nodes of the tree, here please include the terminal nodes.
Do *not* make any attempt to reduce the resulting tree to a ROBDD, as sometimes happens in `apply` implementations. [*6 marks*]

ii. How can computation of the result of `apply` be made more efficient? [*2 marks*]

Answers

1. (a) The desired FSM diagram is:



For each state there is an implicit transition out of the state and looping back whenever the input variable values do not satisfy any of the explicit transitions out of the state.

Marking guide: 3 marks for diagram, 1 mark for sensible definition of implicit transitions

(b)
- `G !(alarm_sounding & state = off)`
- `G (set & G !reset -> F (state = live))`
- `G (alarm_sounding -> (alarm_sounding W reset))`
  `W` here is the weak until operator. Solutions might instead make use of the release operator `R` or could phrase `W` in terms of strong until (`U`) and `G`.

(c) The issue is that, as stated the property expresses nothing about the behaviour of the timer. We need to make an assumption that when the timer is left to run and is not reset, it eventually times out. For example

```
G (G run_timer -> F timeout)
```

Marking guide: 2 marks for the diagnosis of the issues, 2 marks for a sensible fix.

2. (a) i. *Not equivalent:* E.g. consider a model with states $\{r, s\}$, initial state r and transitions $\{r \to r, r \to s\}$ where only $r \models \phi$. This does not satisfy first formula, but does satisfy second.

ii. *Equivalent.*

iii. *Not equivalent:* E.g. consider a model with states $\{r, s, t\}$, initial state r and transitions $r \to s, s \to s, s \to t, t \to t$ where only $t \models \phi$. This satisfies first formula, but not second.

Marking guide: 2 marks each for first two, 3 marks for third.

(b) i. The *language of an LTL formula* is the set of infinite sequences of states that satisfy it. The *language* of a *transition system* $\mathcal{M}$ is the set of infinite sequences of states that it generates.

ii.

$$\mathcal{L}(\phi) \subseteq \mathcal{L}(\mathcal{M})$$

can be phrased as

$$\mathcal{L}(\mathcal{M}) \cap \overline{\mathcal{L}(\phi)} = \emptyset \quad .$$

Observe $\overline{\mathcal{L}(\phi)} = \mathcal{L}(\neg\phi) = \mathcal{L}A_{\neg\phi}$, where $A_{\neg\phi}$ is a Büchi Automaton accepting $\mathcal{L}(\neg\phi)$ (such an automaton always exists).
And note that, for a suitable notion of *composition* $\mathcal{M} \otimes A$ of a transition system $\mathcal{M}$ and BA A, we have that

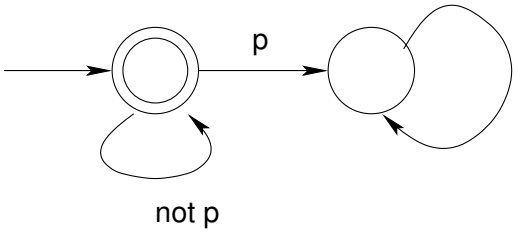
$$\mathcal{L}(\mathcal{M} \otimes A) = \mathcal{L}(\mathcal{M}) \cap \mathcal{L}(A)$$

We can then check $\mathcal{L}(\mathcal{M}) \cap \overline{\mathcal{L}(\phi)} = \emptyset$ by checking

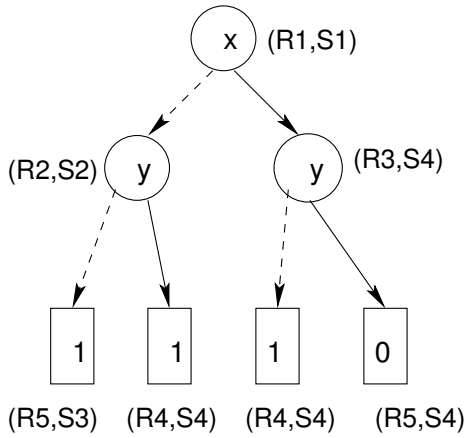
$$\mathcal{L}(\mathcal{M} \otimes A_{\neg\phi}) = \emptyset$$

using a variation on fair CTL model checking.

iii. We could use a Büchi automaton for $\neg Fp = G\neg p$:



(c) i. The resulting binary decision tree is:



ii. In general the algorithm makes recursive calls on the same pairs of nodes more than once. It can be made more efficient by *memoizing*, caching the results of such calls so the results do not have to be recomputed.